An SDN-Supported Collaborative Approach for DDoS Flooding Detection and Containment

Tommy Chin
Rochester Institute of Technology
Rochester, New York, USA
txc9627@rit.edu

Xenia Mountrouidou
Wofford College
Spartanburg, SC, USA
mountrouidoup@wofford.edu

Xiangyang Li
Johns Hopkins University
Baltimore, Maryland, USA
xyli@jhu.edu

Kaiqi Xiong
Rochester Institute of Technology
Rochester, New York, USA
kxxics@rit.edu

***Abstract*—Software Defined Networking (SDN) has the potential to enable novel security applications that support flexible, on-demand deployment of system elements. It can offer targeted forensic evidence collection and investigation of computer network attacks. Such unique capabilities are instrumental to network intrusion detection that is challenged by large volumes of data and complex network topologies. This paper presents an innovative approach that coordinates distributed network traffic Monitors and attack Correlators supported by Open Virtual Switches (OVS). The Monitors conduct anomaly detection and the Correlators perform deep packet inspection for attack signature recognition. These elements take advantage of complementary views and information availability on both the data and control planes. Moreover, they collaboratively look for network flooding attack signature constituents that possess different characteristics in the level of information abstraction. Therefore, this approach is able to not only quickly raise an alert against potential threats, but also follow it up with careful verification to reduce false alarms. We experiment with this SDN-supported collaborative approach to detect TCP SYN flood attacks on the Global Environment for Network Innovations (GENI), a realistic virtual testbed. The response times and detection accuracy, in the context of a small to medium corporate network, have demonstrated its effectiveness and scalability.**

Keywords*—*Denial of Service, SDN, Intrusion Detection, Threat Containment

I. Introduction

Software Defined Networking (SDN) provides the unique capability to separate the data and control planes. To this end, Application Program Interfaces (APIs) such as OpenFlow have been developed to manage and forward data between devices called controllers and network devices that can be either hardware or software based. The software-based switches are typically implemented through Open Virtual Switches (OVS). OVS controllers have complete overview of the network, establish new data flows, and gather various traffic statistics. This wealth of information combined with the flexibility to duplicate, redirect, or terminate flows makes SDN a strong candidate for novel solutions that invoke services and reconfigure security devices on demand.

Recently, SDN has attracted attention for its potential to provide new operational security models. We have categorized relevant studies into two groups: (i) security for SDN to ensure protection of control plane traffic, infrastructure, and applications; and as in this paper, (ii) SDN-supported security services that aim to develop new attack detection and mitigation schemes. So far, the majority of SDN-supported security studies simply propose a conceptual vision of what SDN should be capable to support in a new security service and what the architecture should resemble. A few studies, which actually have experimented with their proposed ideas, are limited in realizing SDNs capabilities. The best arguments these studies have made are that SDN provides more data for attack detection and supports traffic reshaping.

In this study we explore a collaborative approach of attack detection and containment that is unique to SDN. We employ a system of sensory Monitors distributed over a network and attack Correlators residing with SDN controllers at OVS. A Monitor is sensitive and lightweight with the ability to quickly detect anomalies in network traffic. An attack Correlator, alerted by the Monitor, collects more evidence with the help of a controller, to verify pertaining attack signatures in order to conform (or reject) the suspicion. Consequently, the controller coordinates further actions to either update normal traffic profiles in order to avoid future false alerts or to mitigate effects of an attack. We demonstrate the capability of this scheme against TCP SYN flood attacks with the help of Global Environment for Network Innovations (GENI).

II. DDoS Detection and its Big Data Challenge

Distributed Denial of Service (DDoS) attacks have been a growing problem for computer networks and Internet users. They utilize a variety of techniques of flooding, amplification, protocol exploiting, and malformed packets. DDoS attacks pose many significant challenges to current detection and correlation solutions:

- Users generate large volumes of network traffic constantly. Even if a network packet sniffer is able to get hold of every bit of data, it is impossible to do deep inspection on every network packet on the fly.
- The ability of an Intrusion Detection System (IDS) is limited by the information that is available to it. For example, classical attack evasion problems arise when the victim and the IDS have different interpretations of network packets.
- DDoS attacks become more advanced with the use of zombie hosts and reflectors to hide the attacker's traces. Malicious network packets look very similar to normal traffic.

Network threat analysis is a typical big data problem that attracts increasing interest. For example, Bou-Harb, Debbabi,

and Assi [1] have studied the coordination patterns of large-scale probing incidents for behavioral analysis. Camacho, et al. [2] use various statistical transformations to reveal anomalies in firewall log data. However, these studies analyze the traffic data as a whole and many of them do not combine the data on multiple scales, e.g., individual network packets as well as their aggregation. It would be a challenge to implement these solutions for real-time, accurate network attack detection.

This study is related to a few efforts focused on developing a distributed IDS architecture to process a large volume of information. For example, one distributed IDS system uses autonomous agents to monitor security-related activity within a network in [3]. The EMERALD [4] system tries to deal with large-scale network using a hierarchy of Monitors. Topallar, et al. [5] are employing both anomaly detection and signature recognition in a hybrid approach. The novelty of our method that sets it apart from the above studies relies on taking advantage of unique SDN capabilities.

III. SDN-Supported Security

Inspired by SDN flow-based rules, SENSS [6] provides an interface for security attack detection initiated by a victim through queries, where Internet Service Providers (ISP) collaborate to trace back a DDoS attack. In [7]], the authors study entropy-based anomaly detection on flow statistics as well as attack mitigation with the help of OpenFlow and sFlow. In [8], the authors present a lightweight DDoS flooding attack detection design using a NOX controller. In [9]], the authors emphasize additional low-level network information available from SDN that can be used to improve anomaly detection algorithms. Two studies are especially relevant to the challenges and solutions that we consider here. The OrchSec architecture [10] uses decoupled monitors and SDN controllers in order to mitigate different types of attack. The authors have examined several designs including a so-called orchestrator facility. A minimal experimentation over POX and Floodlight controllers is done with Mininet that simply looks for traffic irregularities. This paper mainly predicts a few future research directions. The NICE framework [11] employs an IDS agent to monitor mirrored traffic and to propose potential attack countermeasures. The detection is based on attack graphs of known system vulnerabilities by a so-called attack analyzer at the SDN controller. This system is fairly close to the traditional distributed IDS where a centralized manager aggregates inputs from multiple sensors. OpenFlow only provides a mechanism to support an IDS sensor. Lastly, this paper lacks many specifics in implementation protocols and applications.

In a previous report [12]], we have presented an SDN based security service to detect DoS flooding attacks. We deployed a single Monitor and Correlator to detect a single attacker. In this paper we expand our design to have multiple Monitors and Correlators to prove that it is scalable and can detect distributed attacks. In addition, we perform an extended study of the accuracy and the scalability of the collaborative method as the number of attackers (bots) increases.

In conclusion, we have observed significant problems and challenges in recent studies. First, many studies only present visions on how to take advantage of SDN. In doing so, they do not present specific application scenarios or realistic performance analysis. Second, many studies use SDN hardware switches and facilities in experimentation if any, limited by the support provided by their vendors. The findings are hard to repeat and reuse by other researchers. Lastly, SDN software simulators cannot faithfully fulfill expectations and perform at scale experiments, due to various limitations imposed by their respective software. For example, it is hard for Mininet to support realistic IP-based network sessions.

IV. GENI and SDN

Global Environment for Network Innovations (GENI) has become an emerging virtual infrastructure, which leverages networked computing technologies for experimentations at-scale [13]. GENI is an NSF-sponsored infrastructure. Heterogeneous GENI resources permit users deep programmability throughout the network and are shared among multiple users. The marriage of GENI and SDN gives experimenters control of network packet forwarding within their slices, permits GENI aggregate operators a flexible mechanism to create virtual network slices for experiments, and provides system administrators the opportunity to allocate network traffic into specific slices with appropriate authorization control. As an SDN instantiation, OpenFlow is used to forward network traffic in GENI experiments.

V. A Collaborative Attack Detection and Containment Approach

This study describes our new proposed approach and experimentation on GENI in the context of a campus or corporate SDN network that faces SYN flood attacks. A limited number of computer nodes are connected by multiple OVSs. Multiple attackers target their flooding attack packets at one victim node. In the threat model, these rogue nodes (botnet or zombie hosts) can launch attacks on the same network segment as the victim or from a different segment.

Our investigation includes three generic tasks: first a Monitor raises an alert when an attack occurs; second, informed by the alert, a network attack Correlator further evaluates the situation and if needed, identifies the attackers location and the attack path; and last, if applicable OVS controllers take mitigation actions to block attack traffic or reroute attack packets. We place an emphasis on a distributed architecture, consisting of Monitors and Correlators, while devising a communication protocol for their seamless collaboration. Thus, our strategies and procedures are generic enough to be applicable to other attack scenarios or topologies. However, in a large network with many Monitors and Correlators, requirements to optimize these detection and mitigation actions become significant research issues, which we will report in future studies.

A. Key Observation of DDoS Attack

In a DDoS attack (and many others) multiple features/symptoms of different characteristics together construct its unique signature. Consider the TCP SYN flood attack studied here. One pattern is a surge of SYN requests since this attack tries to deplete network resources in the connection buffer on the victim computer. We can set a control threshold for the Monitor, based on a statistical traffic baseline to quickly raise an alert when such an increase occurs. Unfortunately, a

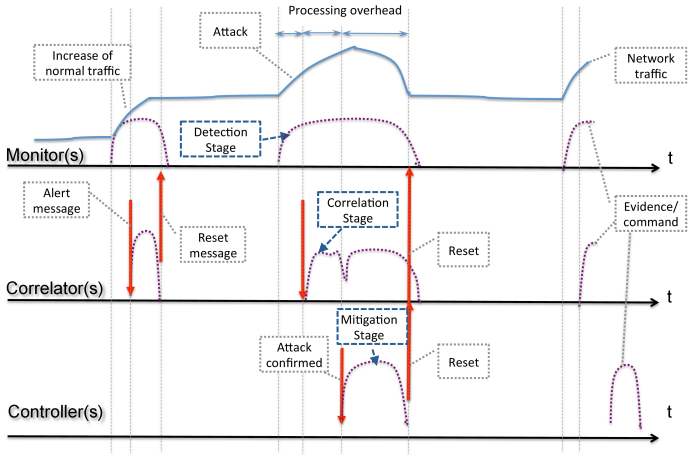


Fig. 1: A pulse-like detection and correlation work cycle, where Monitors distributed over the network raise an alert for a traffic irregularity. Consequently, Correlators are triggered for active evidence collection and forensic correlation with the support of flow forwarding information. If the attack is confirmed, SDN controllers take actions for mitigation.

normal network traffic shift can also increase SYN requests. Thus, this pattern alone is not unique to SYN flood attack. Relying on just this symptom would likely generate false alarms.

Another symptom of this attack is spoofed source IP addresses observed in certain SYN flood requests. These are necessary to prevent connections from being closed immediately. To tell the fake IP addresses apart from legitimate ones requires adequate knowledge of the network topology and configuration. Fortunately such knowledge is available to SDN controllers. Note that in this study we do not consider the situation where non-spoofed IP addresses are used. If needed, we can adjust our method to consider additional symptoms such as incomplete TCP handshakes.

B. A Collaboration of Monitor, Correlator, and Controller

As shown in Figure 1, this collaborative approach deploys three elements. The Monitors, distributed over a computer network, constantly observe the network traffic for any anomalies. The Correlators residing at OVSs respond to the alerts from Monitors on demand. The SDN Controllers themselves take actions to modify the network flows in attack mitigation. The Monitor can employ different anomaly detection algorithms to flag a range of potential attacks. In our design, an anomaly detection engine has two important advantages. First, it is based on a normal behavioral profile, i.e., an expected baseline that can be a statistical abstraction of normal traffic. Therefore, the Monitor does not need to process individual network packets. This helps lowering the computational cost. Second, the detection method is robust since it is based on recognizing deviations from the established baseline, e.g., surge of TCP SYN packets that exceeds a control threshold. Therefore, it can quickly flag any irregularities in network traffic, even new types of attack, at the cost of generating false alerts.

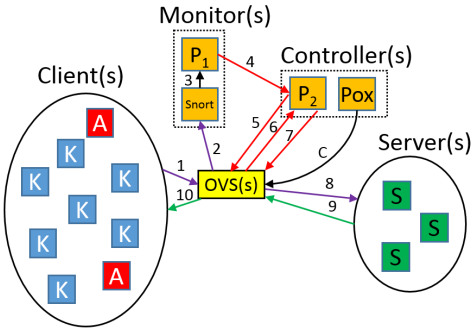


Fig. 2: Logical Architecture Design.

Once an alert is received, a Correlator immediately takes actions based on the type of alert. Often, it is necessary for the Correlator to first collect additional packets and data for pertinent information. After close examination of the evidence, if the Correlator cannot identify the attacker(s), this may indicate simply an increase of normal traffic. Then a reset command is issued to the reporting Monitor(s). The normal traffic profile is updated as necessary to establish a new baseline. Note that the attack may be originated from a network segment different from where the Monitor is or from several network segments simultaneously. Multiple Correlators need to communicate with each other to access related OVSs to reveal these attackers and generate insights of the path of the attack traffic.

At last, confirmation of attack presence in turn triggers attack containment actions as planned. These actions can be executed by SDN controllers and may range from dropping the attack packets, to deploying honeypots to trick the attack for more evidence, to dynamically reconfiguring the network and reshaping the traffic.

VI. System Architecture

In Figure 2, node A is defined as an attacker, K is a legitimate user, and S is a server for a client-server model, where A and K are the clients. Network link (2) is configured as a mirroring port, through which traffic will be transmitted unidirectionally from the OVS to the Monitor. Thus, the Monitor oversees all the traffic that enters the network. The configuration of (2) is set as unidirectional in order to prevent looping mechanisms based on TCP communication in which devices would transmit reset (RST) flagged packets if the traffic was received on the incorrect recipient. Pox within the Controller is using network link (C) for normal flow injection to the OVS, providing SDN-based communication to establish new flows, drop specific flows, etc. Lastly, P_1 and P_2 are programming scripts that are used to communicate between the Monitor and the Controller. The communication protocol is shown in Figure 3. To fully express the architecture and communication protocol, we provide a description as follows:

As A and K communicate with the network (1), the information is forwarded to both the Monitor (2) and the Server (8). After the Monitor receives the traffic (2), the information is processed with Snort, an Intrusion Detection System (IDS). If Snort raises an alert, the information is transmitted (3) to P_1 in order to inform (4) the Controller. The communicator

algorithm P_1 running at the Monitor is listening continuously for alerts in real time. It is lightweight and manages alerts and additional information of relevance. Once an alert is issued by the IDS, P_1 notifies the associated Correlator using a simple attack flag for a specific type of attack. Then the Monitor collects a minimum amount of information related to the attack and sends it to the Correlator. Specifically, for SYN flood attacks, P_1 sends out a number of source IP addresses found in SYN packets.

When the Correlator receives the alert from the Monitor (4), it issues a query of the OVS flow table (5) and retrieves it (6) for correlation. To correlate the information between the Monitor's alert from (4) and the flow table information from the OVS received on (6), P_2 compares the IP address information to the flow table. In a spoofed IP address scenario, where the IP address is falsified as another, an additional comparison occurs. When a node initiates communication to the network for the first time, P_2 will associate that client to their transmitting interface on the OVS by using a database, i.e., original snapshot of the flow table. When a spoofed IP address occurs, P_2 will use the snapshot information as one factor in the correlation. If the Correlator discovers that the IP address transmitted by the Monitor (4) originated from a port that is now assigned to a different IP address than that in the original database (snapshot), it can conclude that the traffic is malicious.

The correlation occurs by examining the number of IP addresses that originate from the transmitting OVS port. We have assumed under a DDoS attack model that the source IP address will be randomized for each packet sent and that the Correlator is not compromised. In P_2, a decision tree is used to drop any future flow of traffic from A on (1). The decision is then transmitted (7) to the OVS. If it was determined that the traffic sent from (1) was deemed malicious, prevention occurs by the action on (7) and the traffic will never reach the S on (8). If the traffic was deemed of good intent, the information is transmitted back to the client (9, 10). As (9) occurs, traffic is transmitted on both (2) and (10) per the design of a mirror port.

If P_2 does not detect spoofed IP addresses, then the Monitor's alert is a false alert. This may be caused by the sensitivity of the IDS and/or the increase of normal traffic. A corrective action can be taken to change the alert threshold of the IDS. Additionally, those TCP SYN flooding attacks that do not use spoofed IPs may cause this. In this case we can consider additional symptoms, such as counting incomplete TCP handshakes.

VII. Experimentation

The implementation on the GENI cloud infrastructure offers the opportunity to validate the assumptions of our design as well as to demonstrate the efficiency and accuracy of our collaborative scheme. In Figure 4 we show the topology of the implementation. Three local networks are connected through a backbone OVS and each network includes its own OVS. There are four OVS switches with their corresponding Monitors, and Controllers. Due to one limitation of GENI, each OVS has only eight available network interfaces. Thus, there are five regular host nodes with the rest of the interfaces reserved for

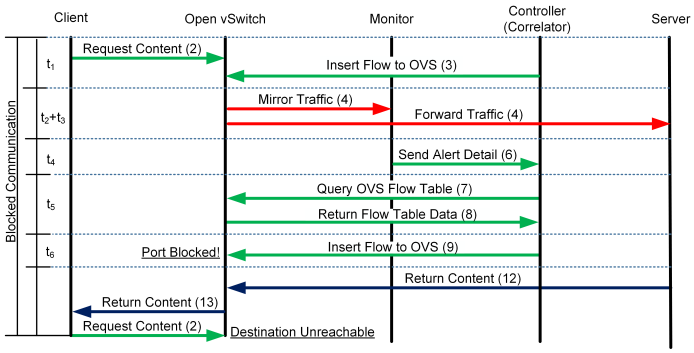


Fig. 3: The communication protocol between the Monitor, the controller (Correlator), and other elements.

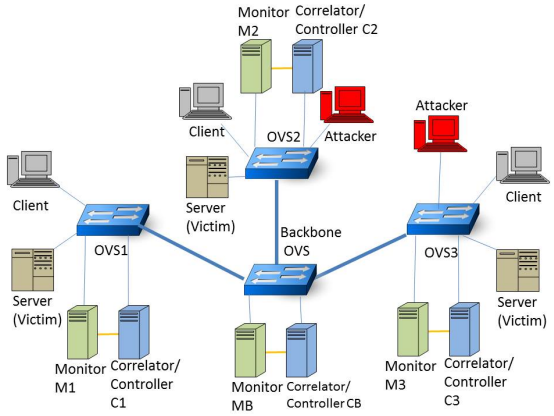


Fig. 4: The implementation topology on GENI that consists of three OVS switches managing three network segments.

a Monitor, a Controller, and an up-link to the backbone OVS. The host nodes, Controllers and Monitors are implemented using XEN VMs that run Ubuntu Linux 64 bit, with Intel(R) Xeon(R) processors at 2.10GHz. The traffic generators create normal traffic with iperf, a commonly used network testing tool that measures performance, to the full capacity of the network bandwidth. We used a tool called hping3 to implement TCP SYN floods with IP address spoofing. Attack traffic is commonly characterized by a rate of a few thousands of SYN packets per second.

The Correlator node is running the correlation algorithm P_2 and the Pox Controller that controls the OVS. Snort IDS is running on each Monitor node. After regular traffic analysis we have devised a Snort rule that is based on a control threshold of the intensity of TCP SYN packets arrival rate. This threshold is equal to the regular traffic rate plus one standard deviation. If the threshold is exceeded, the IDS raises an alert and communication is initiated between the Monitor and the Correlator. The communication protocol between the Monitor and Correlator, P_1, as well as the correlation algorithm, P_2, are implemented in Python using socket programming. We analyze the performance of the collaborative detection by itemizing it to individual intervals, in order to focus on bottlenecks for useful insights. Specifically, we measure the performance using the total duration of detection and mitigation, starting from the moment that the attack is launched until the moment the mitigation scheme is fully in effect. As shown in Figure 3, the duration is calculated using a sum of time intervals: $\sum_{i=1}^{6} t_i$,

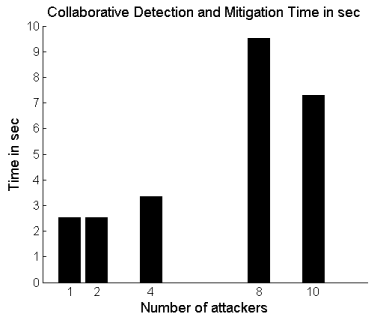


Fig. 5: Total time (seconds) for detection and mitigation for different numbers of attackers.

where t_1: the time elapsed for the mirrored traffic to reach the Monitor; t_2: the constant time, i.e., 1 second, required for the Monitor to process TCP SYN packets and raise an alert based on a specific threshold; t_3: the time required for the Monitor's alert and all the relevant information to reach the Correlator node; t_4: the time required for the Correlator to process the Monitor's alert; t_5: the time required for the Correlator to query the OVS switch; and t_6: the time required for the Pox Controller to transmit proper mitigation actions to the OVS. These time intervals depend either on the network speed, e.g., t_1, t_3, and t_5, or a machine's processing power, e.g., t_4, or both, e.g., t_6. The time reported here is the average time measured on all the Monitors and Correlators involved in a specific scenario with multiple attackers.

Since attack traffic is considerably higher than regular traffic, it affects substantially the performance of the collaborative detection and mitigation method. Thus, we have tested the scalability of the collaborative scheme by increasing the number of attackers, from a single attacker (DoS flooding) up to ten zombie hosts (Distributed DoS). Each scenario is executed ten times. The zombie hosts may be in the same network of the victim or a different network. In the future we plan to investigate how the location of the attackers affects the performance of our scheme. In addition to attack traffic, regular traffic flows are sent to the server at the same time from several host nodes in the topology. Although the number of attackers is limited in the topology, the attack and regular traffic volume is high and comparable to realistic attack scenarios.

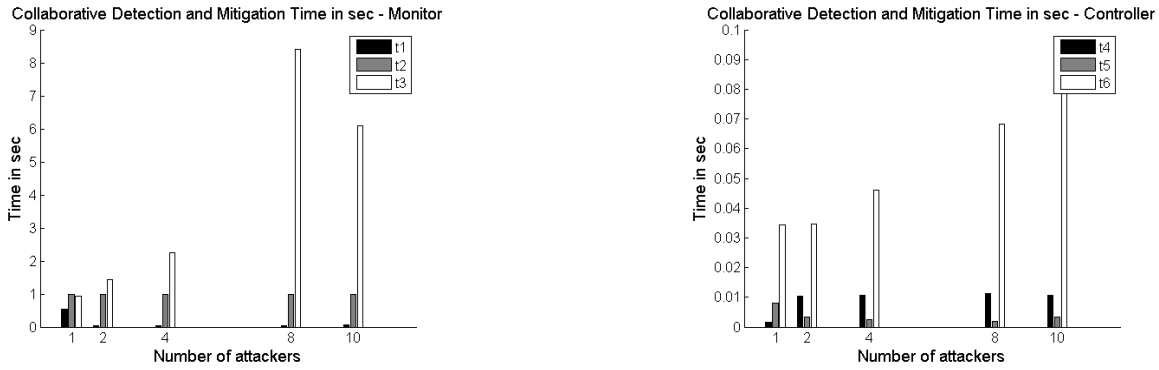
Figure 5 shows that the total time of the collaborative scheme detection and mitigation scales well as the number of zombie hosts increases. The total time from the beginning of the attack until the mitigation action ranges from 2.5-9 seconds and increases almost linearly with the number of the attackers. Surprisingly, in the scenario of ten zombies, the total time to detect and mitigate actually drops. We attribute this to the high volume of SYN packets that immediately triggers the alert mechanism.

Figures 6a and 6b show the individual time intervals that affect the performance of the synergistic detection and mitigation scheme. Figure 6a reveals that the performance bottleneck for the Monitor is the time needed to collect relevant information about the attack, in our case IP addresses, and send it to the Correlator, t_3. Figure 6b on the other hand shows the performance bottleneck for the Correlator is the time required to configure the switch and drop the malicious flows, t_6. According to Figures 6a and 6b the communication of pertinent information between Monitor and Correlator is the dominating performance factor of the collaborative detection and mitigation scheme. This emphasizes the trade off between performance and accuracy; one has to choose between additional communications that increases the accuracy of the detection mechanism or a fast and less accurate alert system.

The accuracy of the collaborative mechanism is evaluated extensively regarding false alerts and false correlation. False positive Monitor alerts are caused when the alert threshold is too low and regular traffic is mistaken for an attack. On the other hand, false negative alerts occur if the Monitor threshold is too high, therefore attacks are not flagged. Furthermore, the Correlator may cause false positive and negative results. Specifically, a false positive correlation occurs if there is a false alert by the Monitor and consequently the Correlator confirms the existence of an attack. False negative correlation occurs when the Correlator does not have enough evidence to find the malicious flow, even though the Monitor has raised a true positive alert. Note that the Correlator's actions completely depend on the Monitor triggering alerts. The Correlator will take no action otherwise.

We use Receiver Operating Characteristic (ROC) graphs as shown in Figure 7 to evaluate the accuracy of our method. We performed the ROC experiments using repetitive attack and burst traffic instances. We define the burst (or flash crowd) traffic as high rate traffic, i.e., 190% of regular traffic rate, sent to the victim node for a short time interval. For each data point of the ROC curve we ran fifty attack and fifty burst instances, using a different Monitor alert threshold value. The Monitor alert threshold value is initially equal to the regular traffic rate plus one standard deviation. We increased this threshold by 30% every time we ran set of fifty attack and fifty burst instances to create another ROC data point.

During the ROC experiments the victim is located in network 1 and the attackers are located in networks 2, 3. Each network includes one attacker node and two regular traffic nodes that send requests to the victim in network 1. The attack traffic to network 1 is sent through the up-link to the backbone. Monitor M1 raises alerts; however Correlator C1 cannot block the attack since it cannot correlate the incoming attack traffic without the help of Correlators C2, C3. For this reason, we do not include ROC graphs for M1 or C1. Our future work will include a communication protocol between Correlators for augmented collaboration of IDS elements over different network segments. We measure the accuracy of Monitors M2, M3 as they collaborate with their corresponding Correlators C2, C3. Figure 7 shows that the Correlators C2, C3 have zero false positive rate. Specifically, when monitors raise false positive alerts the correlators always indicate the ground truth, i.e., that there is no attack traffic. The Monitor M2 appears to have a high positive rate for any alert threshold, on the other hand M3 has relatively lower true positive rate at 80% at its best. This may be attributed to network delays in network 3 that slowed down the attack traffic for lower SYN request intensities and caused M3 not to raise alerts in a few cases. Even though both Monitors have high false positive rates, the Correlators correct this problem by always reaching the ground truth after correlation of the OVS flow tables.



(a) (b)

Fig. 6: (a) Monitor performance, t_1: time elapsed for the mirrored traffic to reach the Monitor; t_2 (1 second): constant time the Monitor needs to process packets and raise an alert; and t_3: time elapsed for the Monitor's alert to reach the Correlator. (b) Controller and Correlator performance, t_4: time elapsed for Correlator to process the Monitor's alert; t_5: time elapsed for Correlator to query the OVS switch; and t_6: time elapsed for controller to transmit the proper mitigation command to the OVS.

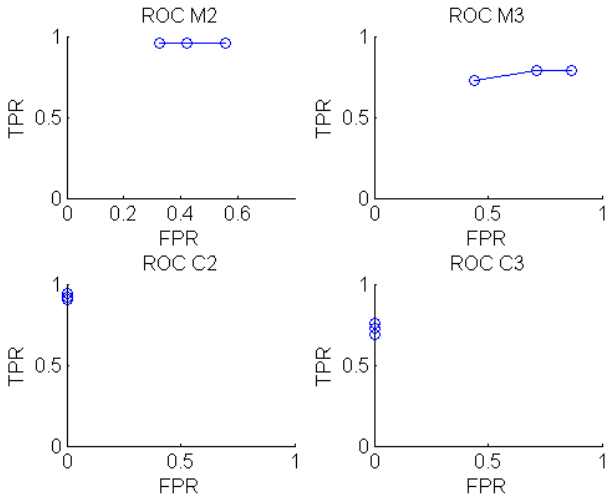


Fig. 7: Receiver Operating Characteristic for Monitor in network 2 (M2), Monitor in network 3 (M3), Correlator in network 2 (C2), and Correlator in network 3 (C3)

VIII. Conclusion and Future Work

We have presented the implementation of a collaborative detection and containment mechanism of network attacks. Our approach is unique to use a synergistic monitoring, detection, and mitigation strategy to realize the full capabilities of SDN. The experimentation on GENI has shown that our solution is scalable to process a high volume of traffic and large scale attacks. The alerting, detection, and mitigation in our system are proven robust through experimentation. Furthermore, the total time required for this collaborative system to detect and contain an attack is low. Thus, this solution can potentially be deployed in a real system where such an attack is detected and mitigated in time before legitimate users start to suffer. We are working to apply this collaborative approach to other security applications, including detection and mitigation of covert channels and other attacks. Our goal is to develop a systematic methodology along this line of work.

References

[1] E. Bou-Harb, M. Debbabi, and C. Assi, "Behavioral analytics for inferring large-scale orchestrated probing events," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 506–511.

[2] J. Camacho, G. Macia-Fernandez, J. Diaz-Verdejo, and P. Garcia-Teodoro, "Tackling the big data 4 vs for anomaly detection," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 500–505.

[3] J. Barrus and N. C. Rowe, "A distributed autonomous-agent network-intrusion detection and response system," DTIC Document, Tech. Rep., 1998.

[4] P. A. Porras and P. G. Neumann, "EMERALD: Event monitoring enabling response to anomalous live disturbances," in *Proceedings of the 20th national information systems security conference*, 1997, pp. 353–365.

[5] M. Topallar, *A New Host-Based Hybrid IDS Architecture-A Mind Of Its Own: The Know-how Of Host-Based Hybrid Intrusion Detection System Architecture Using Machine Learning Algorithms With Feature Selection*. VDM Verlag, 2009.

[6] M. Yu, Y. Zhang, J. Mirkovic, and A. Alwabel, "SENSS: Software defined security service," in *Open Networking Summit 2014 (ONS 2014)*. Santa Clara, CA: USENIX, 2014.

[7] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Computer Networks*, vol. 62, pp. 122–136, 2014.

[8] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE 35th Conference on Local Computer Networks (LCN), 2010*, pp. 408–415.

[9] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*. Springer, 2011, pp. 161–180.

[10] A. Zaalouk, R. Khondoker, R. Marx, and K. Bayarou, "OrchSec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions," in *IEEE Network Operations and Management Symposium (NOMS), 2014*.

[11] C.-J. Chung, P. Khatkar, T. Xing, J. Lee, and D. Huang, "NICE: Network intrusion detection and countermeasure selection in virtual network systems," *IEEE transactions on dependable and secure computing*, no. 4, pp. 198–211, 2013.

[12] T. Chin, X. Mountrouidou, X. Li, and K. Xiong, "Selective packet inspection to detect dos flooding using software defined networking (SDN)," in *2015 IEEE International Workshop on Computer and Networking Experimental Research Using Testbeds*.

[13] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, 2014.